

# Tries and suffixes trees

Alon Efrat  
Computer Science Department  
University of Arizona

## Trie: A data-structure for a set of words

All words over the alphabet  $\Sigma=\{a,b,..z\}$ .  
In the slides, let say that the alphabet is only  $\{a,b,c,d\}$   
 $S$  – set of words =  $\{a,aba, a, aca, addd\}$

Need to support the operations

- $insert(w)$  – add a new word  $w$  into  $S$ .
- $delete(w)$  – delete the word  $w$  from  $S$ .
- $find(w)$  is  $w$  in  $S$  ?

•Future operation:  
•Given text (many words) where is  $w$  in the text.

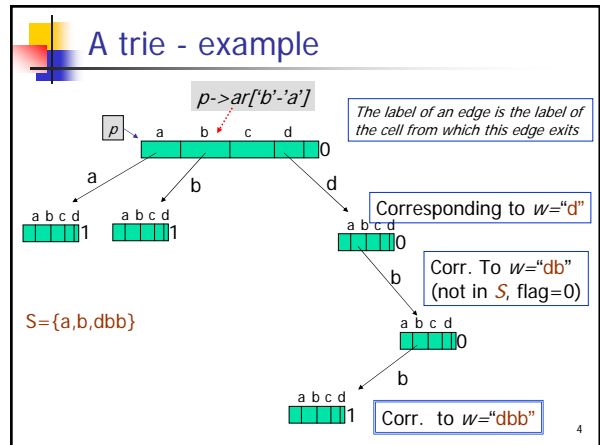
•The time for each operation should be  $O(k)$ , where  $k$  is the number of letters in  $w$

•Usually each word is associated with addition info – not discussed here.

## Trie (Tree+Retrive) for S

- A tree where each node is a struct consist
- Struct node {
  - char[4] \*ar;
  - char flag ; /\* 1 if a word ends at this node. Otherwise 0 \*/

Rule:  
Each node corresponds to a word (which is in  $S$  iff the flag is 1)



## Finding if word $w$ is in the tree

```

p=root; i=0
While(1){
  • If  $w[i] == '\0'$  // we scanned all letters of  $w$ 
    • then return the flag of  $p$  ;
  • If the entry of  $p$  correspond to  $w[i]$  is NULL
    return false;
  • Set  $p$  to be the node pointed by this entry, and set  $i++$ ;
}

```

## Inserting a word $w$

- Try to perform  $find(w)$ .
  - If runs into a NULL pointers, create new nodes along the path.
  - The  $flag$  fields of all new nodes is 0.
- Set the flag of the last node to 1



## Size of suffix tree

Example  $B = \text{"aabab"}$   $S = \{\text{"aabab"}, \text{"abab"}, \text{"bab"}, \text{"ab"}, \text{"b"}\}$

Assume  $n = |B|$ .  
 Total length of all string  $\Theta(n^2)$   
 Size of a node is  $|\Sigma|$   
 So size of the tree is  $\Theta(n^2 / |\Sigma|)$ .

Time to construct the tree  $\Theta(n^2)$

We can save some space.

Example  $B = \text{"aabab"}$   
 $S = \{\text{"aabab"}, \text{"abab"}, \text{"bab"}, \text{"ab"}, \text{"b"}\}$

13

## Suffix tries on a diet

Def: a *shred* is a path from node  $u$  to node  $v$  in the trie, consisting of nodes of outdegree 1 (except maybe the last one) and  $flag=0$ .

Obs: There is a contiguous part of  $B$ , identical to the string the shred represents. We call this part the shred-string.

We store  $B$  itself as an array.

We use a new type of nodes, called shred-nodes, maintain the first ( $id1$ ) and last ( $id2$ ) indexes of the shred-string in  $B$ .

type	a	b	c	d	id1	id2	flag
					7	10	

$B = \text{"c a d b d a a d b d"}$

14

## Suffix tries on a diet - cont

Algorithm for constructing a "thin" trie:

Given  $B$  – create an empty trie  $T$ , and insert all  $n$  suffixes of  $B$  into  $T$  -- generating a trie of size  $\Theta(n^2)$ .

Traverse the tries, and each time that a shred is seen, replace all nodes of the shred with a single shred-node.

15

## Suffix tries on a diet - cont

Clearly the use of shred nodes saves some-but can we prove something ?

Observations: Every the number of leaves of  $T$  is at most  $n$  (every leaf is the end of one prefix)

16

## Suffix tries on a diet - cont

Lemma: Let  $T$  be a tree where each internal node has outdegree 2 or more, and  $m$  leaves. Then  $T$  has at most  $m$  internal nodes.

Back to thin suffix tries:  $T$  does not have **exactly** this property, but it is very close (no long shreds), so a "massaged" lemma still works, so

$\#internal\_nodes \leq \#leaves\_nodes$ ,

But  $\#leaves\_nodes \leq \#suffixes\_of\_B = n$

So the size of the trie is only a constant more than the size of the book.

17